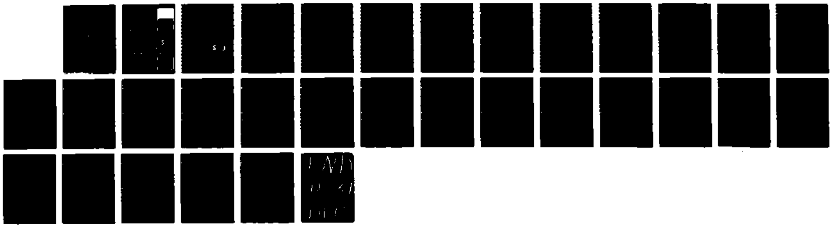
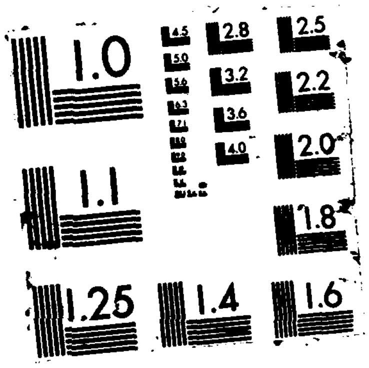
AD-A185 721   OPTIMAL LANE DEPTHS FOR SINGLE AND MULTIPLE PRODUCTS IN   1/1
BLOCK STACKING ST..(U) GEORGIA INST OF TECH ATLANTA
PRODUCTION AND DISTRIBUTION RESE..
UNCLASSIFIED   M GOETSCHALCKX ET AL. AUG 87 PDRC-87-04   F/G 15/5   NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

(12)

Georgia Institute of Technology

OPTIMAL LANE DEPTHS FOR SINGLE AND MULTIPLE PRODUCTS IN BLOCK STACKING STORAGE SYSTEMS

by

Marc Goetschalckx
H. Donald Ratliff

PDRC 87-04

PRODUCTION and DISTRIBUTION RESEARCH CENTER

SCHOOL OF INDUSTRIAL
AND SYSTEMS
ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM
OF GEORGIA
ATLANTA, GEORGIA 30332

(12)

PDRC Report Series 87-04
August 1987

OPTIMAL LANE DEPTHS FOR SINGLE AND MULTIPLE PRODUCTS IN BLOCK STACKING STORAGE SYSTEMS

by

Marc Goetschalckx
H. Donald Ratliff

PDRC 87-04

Report by:
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

This work was supported in part by the Office of Naval Research under Contract No N00014-86-K-0173. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

TABLE OF CONTENTS

LIST OF FIGURES

OPTIMAL LANE DEPTHS FOR SINGLE AND MULTIPLE PRODUCTS IN BLOCK STACKING STORAGE SYSTEMS.

Marc Goetschalckx
H. Donald Ratliff
Georgia Institute of Technology

ABSTRACT

Block Stacking is one of the most common storage methods for warehousing large quantities of palletized or boxed products, when high space utilization is the main concern. Efficient algorithms are currently only available to compute the optimal lane depth for a single product, assuming that all lanes have equal depth.

In this paper, an algorithm is presented to compute the optimal number of lanes and the optimal lane depths for a single product, when the lanes are allowed to have different depths. It is shown that the optimal lane depths follow a triangular pattern. The optimal lane depth pattern is compared experimentally with several heuristic patterns. Several near-optimal and efficient heuristics are identified.

In most warehouses, the lane depth on each side of a single aisle is kept constant for layout and material flow purposes. An optimal algorithm to assign a single product to such a limited number of lane depths is also derived. Based upon this algorithm, a procedure for determining the lane depths and the number of lanes in a warehouse for storing multiple products is developed. If the warehouse is perfectly balanced, then the procedure minimizes the required warehouse area.

(Keywords: Mathematical models; storage and retrieval).

1. INTRODUCTION

Block stacking is one of the most common storage methods for warehousing large quantities of palletized or boxed products. With block stacking no supporting rack structures are used, but the items are stored on top of each other in stacks.. The stacks are placed one after the other in storage lanes and the lanes are placed next to each other perpendicular to the access aisle. Storage and retrieval is done one item at a time, usually with industrial trucks, e.g. fork lift trucks. A conceptual representation of block stacking is shown in Figure 1.

Block stacking is often used for paper goods, household products and appliances and for reserve storage areas, where large quantities per product are stored. In these cases, a high space utilization at low cost is of prime importance.

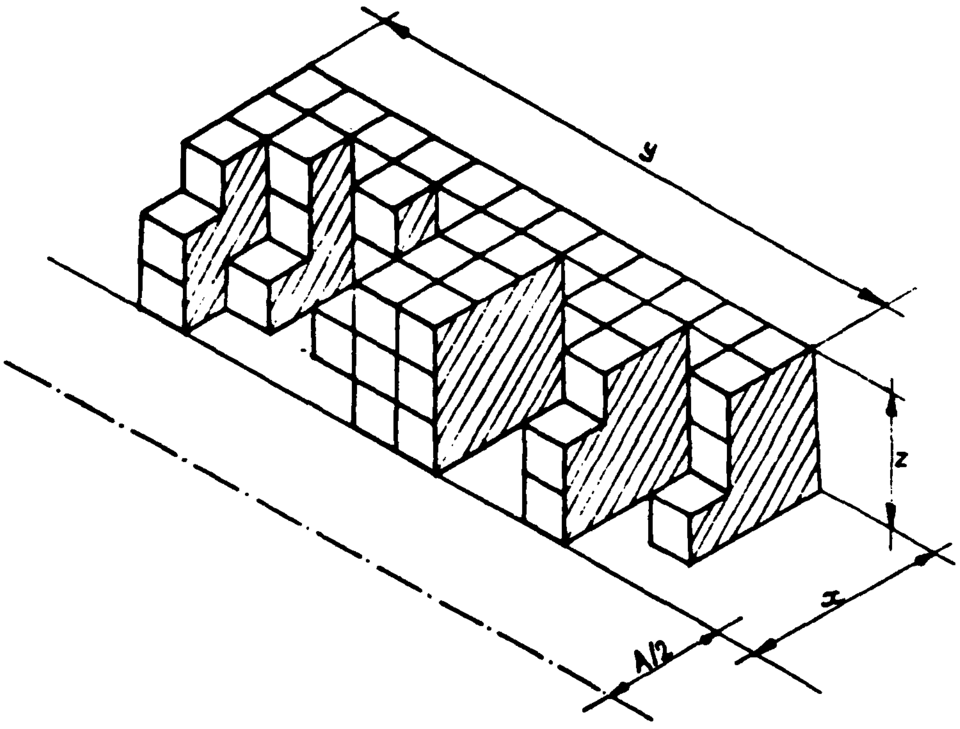


Figure 1. Block Stacking Illustration

Warehouse managers are faced with two important decisions with respect to block stacking. First there is the warehouse design decision, which determines the overall layout and organization of the warehouse. The warehouse layout is relatively constant over time and the design problem must only be solved infrequently. In this paper, it will be assumed that all the items in the warehouse have the same unit load dimensions. The stack height and lane depth can then be expressed in items, rather than in distance units such as feet. The algorithms, which will be presented, can easily be adapted to the case of items with different heights. Furthermore, it will be assumed that the height of the stacks depends strictly on the product load restrictions and warehouse ceiling and is fixed in advance. The stack height will be denoted by z (items). The layout decision thus involves the determination of the depths of the lanes, denoted by x (stacks); the number of lanes of each depth, denoted by y (lanes); and the location of those lanes in the warehouse.

The warehouse manager must also determine the operational policies for managing the warehouse, i.e. the storage and retrieval policies. Most of the time, the items are retrieved on a first in first out (FIFO) basis to prevent item spoilage. The storage policy is the set of rules which determines where each item of an incoming product must be stored. Operational policies can change very quickly and are computed very frequently, thus they must be implemented in real time on the warehouse computer.

Storage policies can be divided into shared and dedicated policies. With shared storage policies it is assumed that the space vacated by an depleted lane is immediately reused by items of another batch (of a different or the same product). With dedicated storage policies a number of lanes is reserved for the units of a particular product. An example of a shared storage policy is random storage and an example of a dedicated storage policy is dedicated storage based on product turnover. It must be observed that if space were to be dedicated to products, then the space utilization would be maximized by making the lanes as deep as possible. The rest of this research will focus on the more space efficient shared storage policies.

Two competing objectives are common in block stacking. On the one hand, the available floor space should be used as effectively as possible, or equivalently, the required warehouse should be built as small as possible. This objective tries to maximize the space utilization of the warehouse layout. On the other hand, the cost for storing and retrieving items from the warehouse should be kept to a minimum. The objective here is to minimize the material handling costs of the storage and retrieval policies. In most cases these two objectives conflict and tradeoffs must be made. For example, deep lanes use the space in front of the lane more efficiently, but require greater care and time for storage and retrieval operations. See Berry (1968) for a further discussion of this issue. This research is primarily concerned with the space utilization objective, since it assumes that large quantities of each product are stored.

For the space utilization objective, the main issue in block stacking systems is determining the lane depth(s) for a product. The aisle space in front of a lane will be denoted by "frontage". Deep lanes require only a small percentage of frontage as compared to total lane space. However, once withdrawal has started from a deep lane, it takes a long time to release that frontage and the empty space in the lane by completely emptying the lane. Short lanes exhibit just the opposite characteristics, in that they require a higher percentage of frontage but the time to release the frontage and the empty lane space is less. Hence, the problem is to determine the lane depth which achieves the best tradeoff between the amount of space it requires and the time this space is occupied.

The following assumptions are typical in a block stacking warehouse. Because of the physical characteristic of block stacking, the storage and retrieval within a lane is last in, first out (LIFO). The retrieval policy for product batches is first in first out (FIFO) to prevent item spoilage. Products and replenishment batches cannot be mixed in a single stack or in a single lane. After some of the stacks in a lane are withdrawn, relocation of the remaining stacks to a shorter lane is not allowed. This minimizes the number of times an item is handled and thus tends to reduce material damage. The withdrawal rate for each product is assumed to be constant over time and the withdrawal quantity is one pallet. Replenishment is assumed to be instantaneous, but a safety stock is allowed.

The following notation will be used throughout the paper. Let Q be the number of items or unit loads to be stored of an arriving product batch. Let A be the aisle width, L the pallet length perpendicular to the a:.le and W the pallet width along the aisle, expressed in distance units such as feet. Both length and width are measured pallet centroid to pallet centroid between adjacent stacks and include all clearances. This corresponds to the "effective length and width" defined by Marsh (1979). Let z be the stack height expressed in unit loads. The different lanes are indexed by n and the lane depth(s) are expressed in stacks and are denoted by a vector x_n. The geometrical definitions are illustrated in Figure 2. Let d denote the constant demand rate for the units of this product. Let I be the current on hand inventory in units of the product, this initial inventory is equivalent to a demand lead time of I/d for the units that have to be stored.

In the next section, the literature on block stacking is reviewed. In Section 3, an algorithm for determining the optimal lane depths (with respect to space utilization) for a single product is introduced. This optimal algorithm is compared in Section 4 with several heuristic algorithms. In Section 5, the case of multiple products is examined and the notion of a perfectly balanced warehouse is introduced. In Section 6, a sequential procedure for designing a warehouse layout and deriving operational policies for block stacking is presented. Finally, the research results are summarized in Section 7.

2. LITERATURE REVIEW

Many authors have discussed the block stacking problem descriptively, e.g. Berry (1968) and Marsh (1979). This review will focus on the literature in which mathematical models for the block stacking problem are presented. One of the most complete treatments of the block stacking problem is given in Matson and White (1984). The most current and extensive review of block stacking to date can be found in Ashayeri and Gelders (1985) which contains many references but does not go into the details of the different methods.

Kind (1965,1975) was the first one to propose a formula for the best unique lane depth for a single product, with respect to the space utilization. Kind proposed the following formula for the single lane depth

$$x = \sqrt{\frac{QA}{Lz}} - \frac{A}{2L}$$

No derivation of this formula is given. This approximate formula was shown by Matson and White (1981) to overestimate the lane depth, but with small relative error, compared to their method.

Kooy (1981) introduced the idea that an optimal warehouse system should always have an empty lane available of the depth required to store an arriving product optimally. This notion is formalized later in this paper to the definition of a "perfectly balanced" warehousing system. Kooy obtained the optimal (multiple) lane depths for a product by complete enumeration over all feasible combinations of lane depths. The number of lanes of each depth required in the warehouse design was determined by running the same complete enumeration program for all the products. No details of the mathematical formulation are given.

The most rigorous treatment to date of block stacking can be found in Matson and White (1981, 1984). They used mostly the space utilization objective. The material handling objective was only considered for a single aisle with lanes of equal depth. For the space utilization objective, they showed that the objective function is non-convex and they derived the optimal unique lane depth for one product by complete enumeration. They also concluded that the objective function, allowing only a single lane depth, is relatively insensitive to small deviations from the optimal depth. They treated the cases with or without safety stock and for different withdrawal patterns. They derived an continuous approximation to the optimal single lane depth, which came very close to the optimal (discrete) result. The formula for the optimal continuous single lane depth is

$$x = \sqrt{\frac{(Q+2I)A}{2Lz}}.$$

The term "lot splitting" is used for the case where multiple lane depths are allowed per product in Matson (1982). An optimal dynamic programming and a heuristic procedure for selecting multiple lane depths out of a set of lane depths for a single product are given. The savings for using 2, 3 or 4 depths were 1.15%, 1.43% and 1.55% respectively, when compared to the optimal single lane depth. It was concluded that multiple lane depths were rarely justified and then at most two different depths were sufficient. No solution times or computational complexity for the dynamic programming procedure or the heuristic are given.

For multiple products. Matson and White identified the influence of the inventory pattern, of the number of different lane depths and of the replenishment schedule on the space utilization and they showed the combinatorial nature of the problem. Lot splitting per product was not allowed. They did not give a solution procedure for this case.

It must be observed that the conclusions drawn by Matson and White are based on the comparison of heuristic solutions to the single product block stacking problem. In the next section, an optimal solution for the single product problem will be derived, which will allow the validation of the above conclusions. If multiple products have to be stored, then their unique lane depths might not match and lot splitting must be used in order to obtain a feasible warehouse layout. This problem is treated in Section 6.

3. OPTIMAL MULTIPLE LANE DEPTHS FOR A SINGLE PRODUCT

In this section an algorithm is presented which will find the optimal multiple lane depths for a batch of a single product with respect to minimizing the total space over time the batch requires during its stay in the warehouse. The lane depths and the number of lanes for each lane depth are determined by the algorithm. It will be shown that the optimal lane depths follow a triangular pattern, with a slope equal to half the aisle width divided by the pallet length.

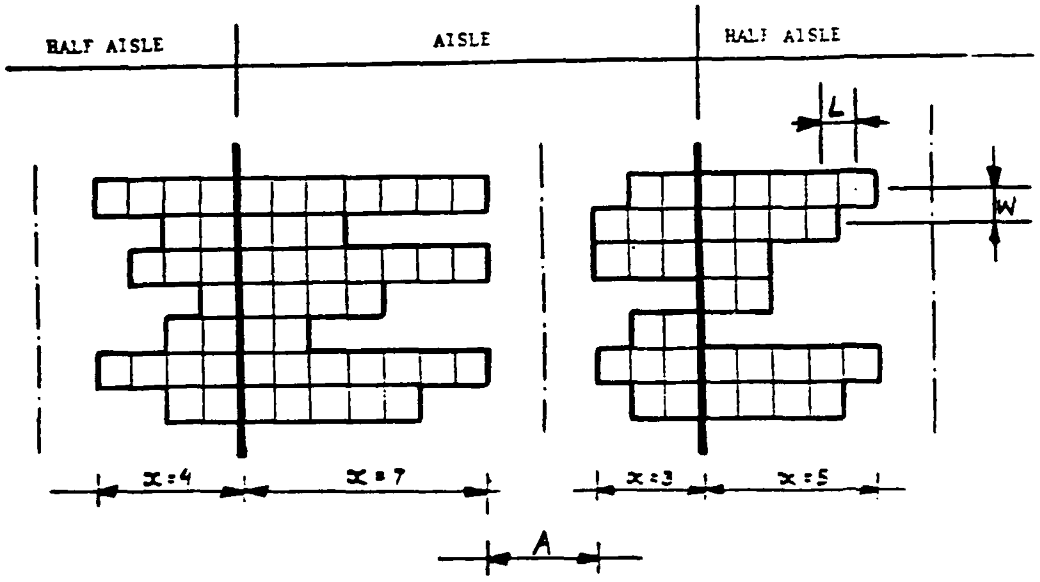


Figure 2. Block Stacking Ground Plan

It is assumed that products are stored back-to-back in the warehouse as illustrated in Figure 2. One half of the frontage is allocated or "charged" to that lane. Assuming that a depleted lane is immediately reused by other units, the basic question is how much space and for how long a particular batch requires. The objective is thus to minimize the total space required over time for the storage of this batch. The dimensions of the objective are $[L^2T]$, e.g. square feet days. Since the total time the batch stays in the system is independent of its storage pattern, the above objective is equivalent to minimizing the average required space for the batch.

Furthermore, it is assumed that the lanes are emptied by increasing index n, i.e. the demand is first withdrawn from lane 1, then from lane 2, etc... If the number of pallets Q is not an integer multiple of the the stack height z, then it is assumed that the first stack in the first lane is not of full height. Similarly, if there are not enough stacks to fill all lanes to their full depth, then it is assumed that the first lane is incomplete.

The optimal algorithm is based on dynamic programming. The states in the dynamic programming formulation correspond to the number of stacks, the stages correspond to the number of lanes (which can be at most equal to the number of stacks). The computational complexity of the algorithm is then equal to $O(P^2)$, where P is the number of stacks. An algorithm is said to be $O(N^2)$ if N is a measure of the size of the problem instance and if the running time of the algorithm does not grow faster than a quadratic function of N. A more rigorous definition of algorithm complexity is given in Aho et al. (1983).

The algorithm was programmed in Pascal on a 8 Mhz IBM AT with 80287 numerical coprocessor. The computation times ranged from less than 0.05 seconds for five stacks to 11.21 seconds for 160 stacks, with an average time of 1.57 seconds. In Section 4, a complete description of the parameters of the experiment is given. These short computation times allow the implementation of this optimal algorithm in real time on small warehouse computers.

In the remainder of this section, notation and formulas are derived which are used in the derivation of the optimal triangular pattern and in the optimal dynamic programming algorithm.

Optimal Triangular Pattern

Let P and p denote the required number of stacks for storing Q and q pallets respectively, or

$$\begin{cases} P = \lceil Q/z \rceil \\ p = \lceil q/z \rceil , \end{cases} \tag{1}$$

where $\lceil x \rceil$ indicates the ceiling function (i.e. the smallest integer value larger than x).

Let x_n denote the number of stacks located in lane n. Let N denote the index of the last lane in which stacks are to be stored. Observe that N is not known in advance. Let r_n denote the number of stacks located in the lanes n+1 through N and, thus

$$\begin{cases} r_0 = P \\ r_n = \sum_{j=n+1}^{N} x_j , \text{ for } n=1..N-1 \\ r_N = 0. \end{cases} \tag{2}$$

Let $S_n(q,r_n)$ be the total space occupied over time by lane n if a total of q pallets are stored in the lanes n through N and if there are a total of r_n stacks, or equivalently r_nz pallets, located in the lanes n+1 through N. The depth of lane n is then determined with

$$x_n = p - r_n . \tag{3}$$

The space required over time by lane n is equal to the product of the space it requires multiplied by the time it occupies that space. Hence

$$\begin{aligned} S_n(q,r_n) &= [W(x_nL+A/2)]\cdot[(I+Q-r_nz)/d], \text{ if } x_n>0 \\ &= 0 \text{ otherwise.} \end{aligned} \tag{4}$$

The space required over time by the all the items in the batch is then

$$S = S_1(Q,r_1) + \sum_{n=2}^{N} S_n(r_{n-1}z,r_n). \tag{5}$$

The optimal dynamic programming procedure assigns the lane depths in a triangular pattern, (i.e. pallets that will remain in the warehouse for a longer period of time are stored in deeper lanes). If this pattern is approximated by a continuous triangle, then the slope of the pattern is equal to A/2L. The lane depth of lane i is then equal to

$$x_i = \frac{iA}{2L}. \tag{6}$$

This can be shown by a reformulation of the problem. Substitution of (1) through (4) in (5) yields

$$S = W\left\{\frac{A(Q+I)N}{2d} + \frac{L(Q+I)}{d}\cdot\sum_{i=1}^{N} x_i - \frac{Az}{2d}\cdot\sum_{i=1}^{N} r_i - \frac{Lz}{d}\cdot\sum_{i=1}^{N} x_i r_i\right\}. \tag{7}$$

Substitution of

$$\sum_{i=1}^{N} x_i = P, \tag{8}$$

$$\sum_{i=1}^{N} r_i = \sum_{i=1}^{N-1}\ \sum_{j=i+1}^{N} x_j = \sum_{i=1}^{N} ix_i - P, \tag{9}$$

and

$$\sum_{i=1}^{N} x_i r_i = \sum_{i=1}^{N-1}\ \sum_{j=i+1}^{N} x_i x_j = (P^2 - \sum_{i=1}^{N} x_i^2)/2 \tag{10}$$

in (7) yields

$$S = \frac{W(Q+I)LP}{d} + \frac{WzP(A-P)}{2d} + \frac{W(Q+I)AN}{2d} + \frac{WLz}{2d}\cdot\sum_{i=1}^{N}(x_i^2 - \frac{Ai}{L}x_i). \tag{11}$$

Taking the partial derivative of (11) with respect to x_i gives

$$\frac{dS}{dx_i} = \frac{WLz}{2d}(2x_i - Ai/L) \tag{12}$$

and setting (12) equal to zero yields (6).

If there is any initial inventory or safety stock, then the dynamic programming procedure and its continuous approximation acts as if this safety stock is stored first in the triangular pattern and the new batch is appended to the safety stock (even though the safety stock is most likely not stored in this pattern). See Figure 3 for an graphical illustration of this pattern. The shaded area corresponds to the safety stock, which is not necessarily stored in this pattern.

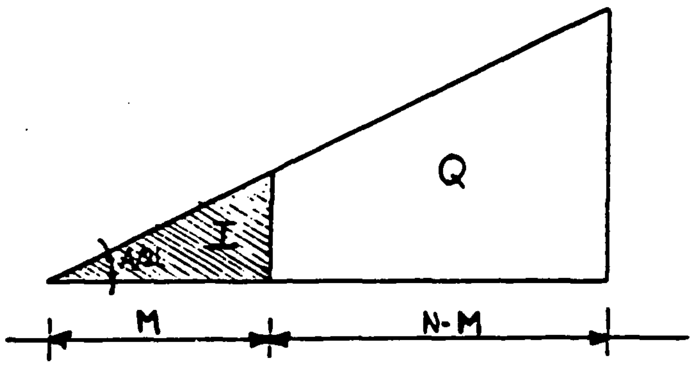


Figure 3. Triangular Lane Depth Pattern with Safety Stock.

For large batch sizes with no initial inventory, the number of lanes in the continuous approximation of the triangular pattern is equal to

$$N = \lfloor 2\sqrt{LQ/Az} \rfloor , \qquad (13)$$

where $\lfloor x \rfloor$ denotes the floor function (i.e. the largest integer smaller than x). Let M be the number of lanes required as if the initial inventory was stored in the triangular pattern, then

$$M = \lfloor 2\sqrt{(IL/Az)} \rfloor . \qquad (14)$$

The number of lanes required for the new batch, when initial inventory is on hand, is then

$$N = \lfloor 2\sqrt{L(Q+I)}/Az - 2\sqrt{(IL/Az)} \rfloor \qquad (15)$$

The above derivation provides an overall pattern for the lane depths and is used in a continuous approximation heuristic described in Section 4. But, because the lane depths can only assume integer values, a dynamic programming algorithm is required to find the discrete optimal lane depths. The required recursion formula will be derived next.

Optimal Dynamic Programming Algorithm

Let F(i,j) denote the minimum amount of space required over time for storing optimally i pallets into the lanes j through N. Consider first the case of locating q pallets from a batch of Q pallets in the last two lanes N-1 and N. The objective is to find F(q,N-1). If all pallets are stored in the last lane, and thus $x_{n-1}=0$, then

$$F(q,N-1) = 0 + F(q,N) = W(pL+A/2)\cdot(Q+I)/d. \qquad (16)$$

For the case where there are pallets stored in the last and next to last lane, then

$$F(q,N-1) = [W((p-x_N)L+A/2)]\cdot[(I+Q-x_N z)/d] + [W(x_N L+A/2)]\cdot[(I+Q)/d]. \quad (17)$$

To find the optimal x_N, the first derivative of (17) with respect to x_N is set equal to zero which yields

$$x_N^* = p/2 + A/4L. \qquad (18)$$

The second derivative with respect to x_N is equal to 2WLz/d, which is positive. Hence, (17) is a convex function of x_N. The integer optimal x_N^* can then be found by evaluating (17) for $\lceil x_N^* \rceil$ and $\lfloor x_N^* \rfloor$. Selecting the integer x_N^* which generates the smaller objective function value of the two yields the integer optimum of (17). Taking the smaller value of (16) and (17) for the two cases (with or without pallets stored in lane N-1) yields the optimal F(q,N-1).

So far the optimal lane depths for the last two lanes have been derived if q pallets have to be stored in the last two lanes. The optimal lane depths for q pallets in the last three lanes can then be found by trying every combination of x_{N-2} and the optimal lane depths in the last two lanes for $r_{N-2}z$ pallets, where $r_{N-2}=p-x_{N-2}$.

In general, the optimal lane depth vector x^* can be found with dynamic programming and the following backwards recursion formula

$$F(q,n) = \min_{x_n=0..p-N+n} \{S_n(q,x_n) + F(x_n z, n+1)\}. \tag{19}$$

The range of x_n is based on the fact that there are N lanes in total and at least one stack must be assigned to each of the lanes $n+1$ through N, i.e. at least N-n stacks are assigned to the lanes $n+1$ through N.

The range of the parameters q, n and x_n can be further reduced based on the following property, which states that the optimal vector x^* must contain non-decreasing elements.

<u>Increasing Lane Depth Property</u>

For any optimal location policy, the elements of the lane depth vector x must have non-decreasing values.

<u>Proof</u>. Assume an optimal lane depth vector x with two adjacent lane depths x_s and x_t such that $t=s+1$ and $x_s > x_t$. Let S be the total space required over time for this storage vector and let R be the total space required over time by interchanging the elements x_s and x_t. Let v be the number of pallets in all lanes before lane s. Then based on (1) through (5) with C equal to the space contribution of all other lanes,

$$\begin{aligned} S &= C + W(x_s L + A/2)\cdot(v+x_s z+I)/d + W(x_t L+A/2)\cdot(v+x_s z+x_t z+I)/d \\ R &= C + W(x_t L + A/2)\cdot(v+x_t z+I)/d + W(x_s L+A/2)\cdot(v+x_t z+x_s z+I)/d \end{aligned} \tag{20}$$

and

$$S-R = (x_s - x_t)AWz/2d > 0. \tag{21}$$

Hence the lane depth vector x cannot be optimal since interchanging the elements x_s and x_t decreases the objective function. By successively repeating this argument the optimal lane depth vector x must have its elements order by non-decreasing value. Q.E.D.

This property further reduces the allowable range of x_n in the recursion formula. If p stacks have to be assigned to the lanes n through N, with $p = \lceil q/z \rceil$, then lane n can have at most $\lfloor p/(N-$

$n+1)\rfloor$ stacks. This is equal to the number of stacks that all the lanes must have if each lane had the same depth. Furthermore, assume that for lane $n+1$ the largest optimal lane depth is equal to M_{n+1} for all possible q. The number of stacks in lane n can never be larger than this value M_{n+1}. Hence, the new upper bound of x_n becomes

$$x_n \leq \min\,(M_{n+1},\ \lfloor p/(N-n+1)\rfloor). \tag{22}$$

The computation of M_{n+1} can easily be incorporated in the backwards dynamic programming recursion. The property also provides the stopping criteria for the dynamic programming recursion. If M_{n+1} is equal to zero, then all the lanes with a smaller index n must also have zero stacks assigned to them and the dynamic programming procedure can terminate. After the algorithm has terminated, N is set equal to the number of non-empty lanes in the optimal solution.

4. HEURISTIC LANE DEPTHS FOR A SINGLE PRODUCT

In this section several heuristic procedures for determining the lane depths are compared with the optimal procedure of the previous section. The objective is to minimize the total space over time a single batch requires during its stay in the system.

Design of Experiments

All the heuristics are compared with the optimal algorithm by the following full factorial experiment. The first factor in the experiment is the batch size expressed as the number of stacks in the atch. This number was set equal to 5, 10, 20, 40, 80 and 160 stacks. The second factor is the ratio of the aisle width and the pallet length (measured perpendicular to aisle, from pallet centroid to pallet centroid). This ratio was set equal to 2, 3, 4, 5, 6 and 7. For a pallet length of 4 feet, this corresponds to aisle widths of 8, 12, 16, 20, 24 and 28 feet. A third factor is the inventory present at the time of arrival as a percentage of the arrival batch size. This percentage was set equal to 0, 20, 40, 60, 80 and 100%. For each algorithm a total of 216 cases were compared. Eight different algorithms were compared for a total of 1728 cases. It must be observed that the demand rate, the pallet width along the aisle and the stack height are invariant factors in the experiment, i.e. they will affect the absolute values of the objective function but not the relative error of the heuristics or their running times.

All heuristics were programmed in Microsoft Pascal (Version 3.32) on an IBM AT with 80287 numerical coprocessor with all debugging options disabled. Running times are for the computation of the lane depths only and exclude all input/output operations. The resolution of the IBM AT internal clock is 0.05 seconds. The relative error in percent of the heuristic algorithms in function of the number of stacks and the A/L ratio are shown in Figures 4 and 5, respectively.

Equal Heuristic

The first heuristic is the equal lane depth heuristic as presented by Matson and White (1984). This heuristic is denoted by "Eq" for Equal in Figures 4 and 5. Instead of allowing multiple lane depths for a single product, this heuristic finds the optimal single lane depth by complete enumeration over the lane depths. The relative error of the heuristic ranged from 0 % to 9.09 %, with an average relative error of 0.76 %. The relative error decreased with increasing number of stacks, increasing A/L ratio and with increasing inventory percentage. The running times ranged from less than 0.05 seconds to 0.11 seconds with an average of 0.03 seconds and can be called negligible.

Continuous Equal Heuristic

The second heuristic is the continuous approximation of the equal lane depth as described in Matson and White(1984). This heuristic is denoted by "CE" for Continuous Equal in Figures 4 and 5. The optimal continuous equal lane depth is given by

$$x = \sqrt{\frac{(Q+2I)A}{2Lz}},$$

The relative error of the heuristic ranged from 0 % to 34.40 %, with an average relative error of 5.17 %. The relative error decreased as the number of stacks increased, but increased with increasing A/L ratio and increasing inventory percentage. The continuous equal heuristic performed worse and worse compared to the discrete equal heuristic for increasing inventory percentage. The running times were always lower than 0.05 seconds and can be ignored.

Triangle Heuristic

The heuristic, which uses the continuous approximation of the optimal triangular pattern, is denoted as the "Tr" for Triangle heuristic in Figures 4 and 5. The relative error of the triangle heuristic ranged from 0 % to 9.23 %, with an average relative error of 0.86 %. The relative error decreases with increasing number of stacks, but is independent of the A/L ratio and the percentage inventory. Except for some exception cases (with small number of stacks), where the relative errors were almost 8 %, the continuous approximation performed virtually the same as the discrete optimal procedure. The running times were always lower than 0.06 seconds and can be ignored.

Pattern Heuristic

It is usually not practical to have a large variety of lane depths in a warehouse, since the lane depth in a single aisle is kept constant for layout and material flow purposes. To incorporate this practical constraint, a modification of the optimal procedure was developed which allowed a limited number of lane depths. In the experiment two sets of different lane depths were compared. In the first set the number of different lane depths was set equal to 6 with values 1, 2, 5, 10, 20 and 40. This heuristic was denoted by "5P" for 5 Pattern. In the second set the number of lane depths was set equal to 6 with values 1, 2, 4, 8, 16 and 32. This heuristic was denoted by "2P" for 2 Pattern. The dynamic programming procedure can be modified to only examine these lane depths.

The relative error of the 5 Pattern heuristic ranged from 0 % to 6.06 % with an average error equal to 0.05 %. The percentage error decreased with increasing number of stacks and with increasing percentage inventory, and decreased slightly with increasing A/L ratio. The running times ranged from less than 0.05 seconds to 4.34 seconds with an average time of 0.65 seconds. This is approximately one third of the time required for the completely unrestricted procedure. The running times increased with increasing number of stacks, but decreased with increasing A/L ratio and decreased sharply with increasing inventory percentage.

The relative error of the 2 Pattern heuristic ranged from 0 % to 20.0 % with an average error of 2.01 %. The percentage error decreased with increasing number of stacks, but increased with increasing A/L ratio and increasing percentage inventory. The running times exhibited the same behavior as in the 5 Pattern heuristic. They ranged from less than 0.05 seconds to 4.67 seconds with an average time of 0.68 seconds.

Different number of allowable lane depths were also tested by eliminating one of the allowed lane depths. There was no large difference in the overall performance as long as there were more than 4 lane depths allowed which formed a (approximately) geometrical series of sufficient range. If some batches have a small number of stacks (such as 5 stacks in the experiment), then a lane depth equal to one must be included in the allowable lane depths in order to achieve low error bounds. Similarly, if the number of stacks is very large then a large lane depth must included to keep the errors low. The error of the 5 Pattern and 2 Pattern behaved differently in function of the percentage inventory and the A/L ratio.

In addition, a 3 Pattern consisting of 6 lane depths 1, 3, 6, 12, 24 and 48 was also investigated. The performance of this pattern was much worse. The relative error ranged from 0 % to 45.29 % with an average error of 6.23 %. This can be explained by the fact that these lane depths are not integer divisors of the number of stacks in the batches. Hence, the lane depth pattern should form an approximately geometric series from 1 to P/4, where P is the largest common batch size (in stacks), and the individual depths should be integer factors of the most common batch sizes.

1 Lane and Q Lanes Heuristic

For comparison purposes, the error for the following two extreme storage patterns was also computed. The first pattern consists of one lane with depth equal to the number of stacks and this heuristic will be denoted by "1L" for 1 Lane. The second pattern has a number of lanes equal to the number of stacks and all the lanes are 1 deep. This heuristic will be denoted by "QL" for Q Lanes.

The relative error of the Q lanes heuristic ranged from 0 % to 283 % with an average error of 122 %. The relative error increased sharply with increasing A/L ratio, increasing number of stacks and increasing percentage inventory. This confirms the rules of thumb that the lanes get deeper when the items stay longer in the system (because of large batch size or large initial inventory) or when the lanes get wider (in order the use the lane space more efficiently). The running times were always less than 0.05 seconds and can be ignored.

The relative error of the 1 Lane heuristic ranged from 0 % to 74.16 %, with an average error of 17.5 %. The relative error increased with increasing number of stacks, but decreased with increasing A/L ratio and increasing percentage inventory. A large number of stacks will be split up over more than one lane. But again the lanes get deeper if the aisles get wider or if there is more inventory. The running times were always less than 0.05 seconds and can be ignored.

Conclusions

From the above experiment it can be concluded that the space utilization objective function for a single product is relative insensitive to the lane depths. Most of the heuristics generate an excellent solution quality with very little computational effort. Based purely on the solution quality and the required computational effort, all heuristics (except the two extreme patterns) and the optimal procedure are considered equivalent. This validates the conclusions drawn by Matson and White.

For multiple product situations, the heuristic which finds the optimal lane depths out of a limited set of allowable lane depths is the most practical one. Its quality and efficiency are excellent. In multiple product warehouses, lot splitting is very important from the warehouse layout point of view. This was not identified by Matson and White. The discussion from now on will focus on this pattern heuristic.

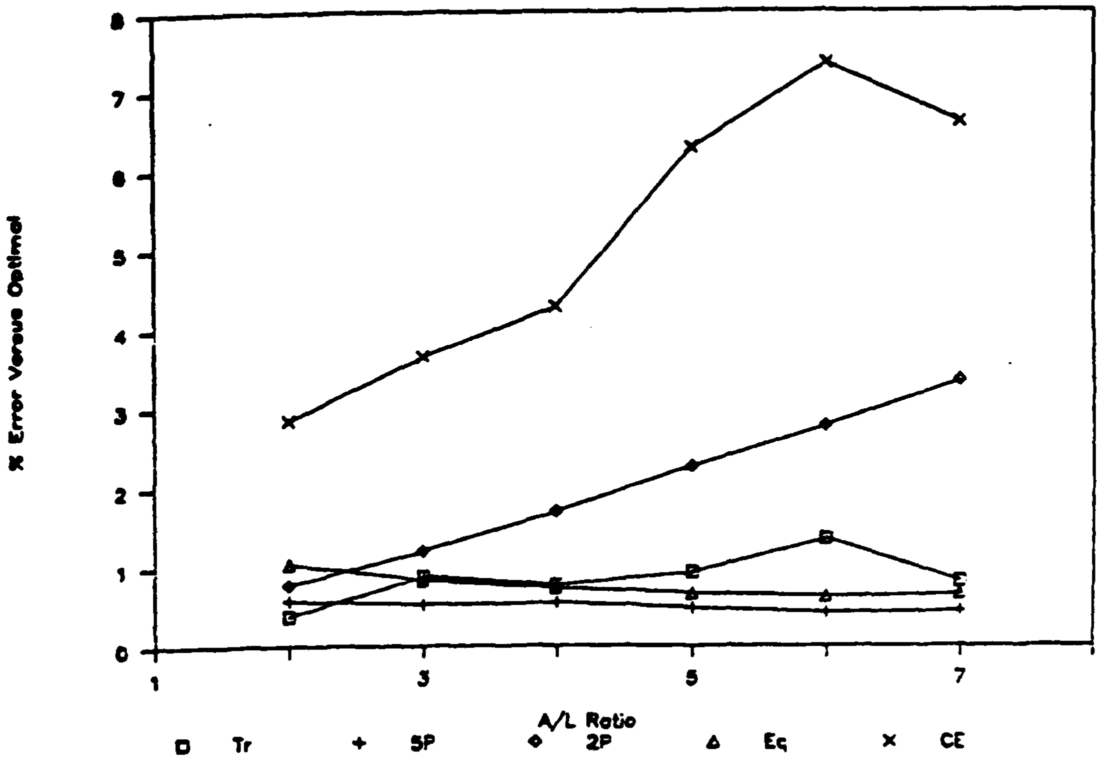


Figure 4. Relative Error of Heuristics versus Optimal Procedure for Different Batch Sizes

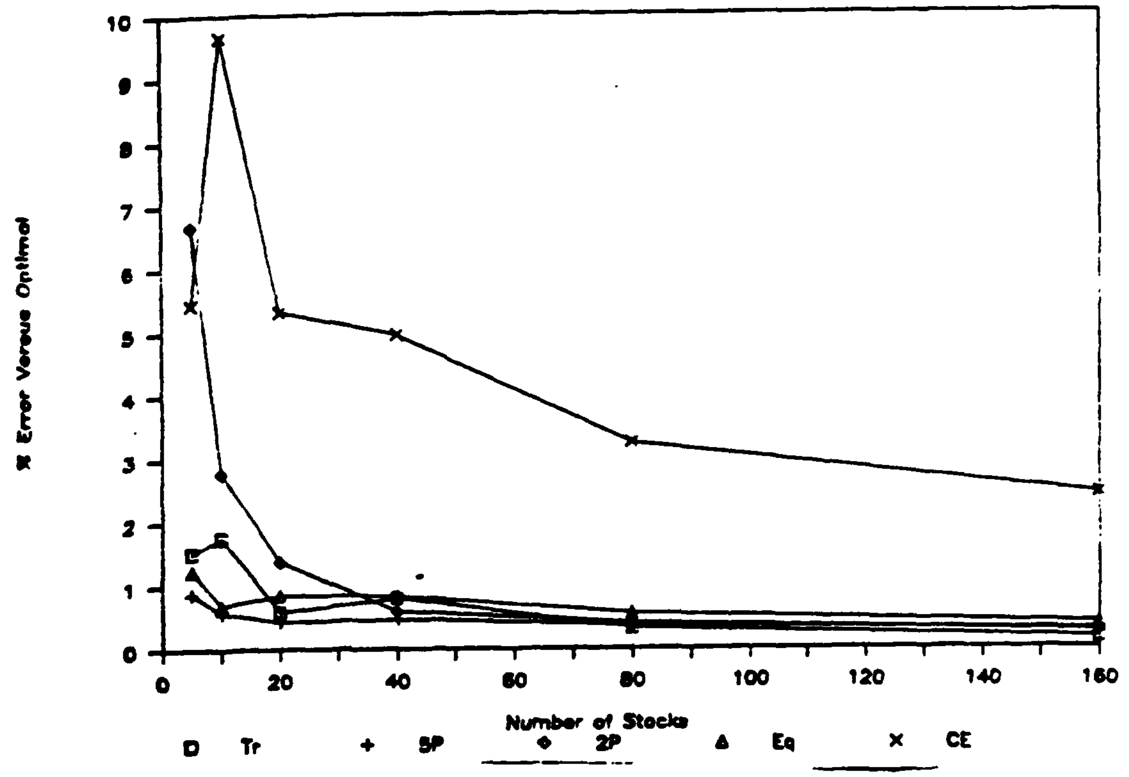


Figure 5. Relative Error of Heuristics versus Optimal Procedure for Different A/L Ratios.

5. LANE DEPTHS FOR MULTIPLE PRODUCTS AND SHARED STORAGE POLICIES

In the above derivation of the lane depths it was assumed that a shared storage policy was used, i.e. as soon as product vacates a lane, this lane is no longer charged to that product. In other words, it is assumed that another batch will immediately occupy the vacated lane. This assumption is unlikely to be satisfied in real life situations.

Three main factors determine how well products can share the space in a warehouse. They are the number of products, the replenishment and withdrawal patterns of the products and the replenishment batch sizes and the safety stocks of the products. For more information on shared storage policies, see Goetschalckx and Ratliff (1984).

Perfectly Balanced Warehouse

Consider the case of a "perfectly balanced" warehouse. A warehouse is perfectly balanced if and only if every time a lane of a certain depth is vacated, a batch arrives which requires a lane of that length for its optimal space utilization. The perfectly balanced condition is not likely to be satisfied in a real warehouse. For perfectly balanced warehouses, a shared storage policy can be formulated which minimizes the required warehouse space.

Minimum Warehouse Size Property

In a perfectly balanced warehouse, the required warehouse space is minimized by computing the optimal lane depths for each arriving batch and by assigning the batch to the empty lanes of the required depth.

Proof. If a warehouse system is perfectly balanced, every batch that arrives can be stored in its optimal depth pattern, since an empty lane of the required depth will always be available, and no lane in the warehouse will ever be empty. Since the each batch is stored optimally and since all products together do not require any more space than the individual products, a perfectly balanced warehouse has the optimal space utilization, or equivalently, the required warehouse size is minimized.

Kooy (1981) expressed similar ideas in a qualitative way. The above definition of perfectly balanced can be relaxed based upon a limited number of allowable lane depths, i.e. substitute the pattern heuristic for the optimal procedure to determine the lane depths for every arriving product. This will be denoted as a "pattern perfectly balanced" warehouse. A pattern perfectly balanced warehouse requires a smaller number of different lane depths and this will make the pattern perfectly balanced condition more likely to be satisfied in real life. Hence if a warehouse is pattern perfectly balanced, then the pattern storage policy will yield an minimum space configuration, i.e. the pattern storage policy is optimal.

If a warehouse is not completely pattern perfectly balanced and if a product arrives and its lane depths are computed using the pattern heuristic and there is no empty lane of that particular length, then a storage lane of the next longer or shorter length can be used. Because of the relative insensitivity of the objective function, the use of this heuristic will not significantly decrease the solution quality.

Observe that the above policy will operate the warehouse very efficiently with respect to required space, but that the warehouse might be very inefficient with respect to material handling cost.

The main disadvantage of the above warehouse configuration is that the number of lanes of a certain depth might not be a multiple of the number of lanes that fit in an aisle. The following section will develop a simple rounding procedure to achieve this and hence will yield a "practical" warehouse layout.

6. WAREHOUSE LAYOUT DESIGN AND OPERATING POLICIES

For a warehouse system in steady state operation, the following set of procedures can be specified, which will make this warehouse operate in near-optimal mode with respect to space utilization. It is assumed that the warehouse is nearly perfectly balanced with respect to the pattern heuristic. This assumption seems reasonable if the warehouse system has many, uncorrelated products and the aggregate inventory is relatively constant.

The procedure consists of three steps. In step one, the required aggregate number of lanes for each lane depth based on a certain depth pattern is computed. In step two, the number of lanes of each depth is rounded to multiples of the number of lanes in an aisle. This determines the number of aisles of each lane depth. The first two steps determine the warehouse design. In the third step, the warehouse storage policy is then specified. The three steps can then be repeated to evaluate different depth patterns.

Warehouse Layout Procedure

1) For all products currently in the warehouse, compute the optimal number of lanes based on a particular lane depth pattern. Sum the required number of lanes by depth for all products. Repeat this computation at regular time intervals and average out the required number of lanes for each depth.

2) Sort these required number of lanes by increasing depth. In order for no lanes of a different depth to be mixed on one side of an aisle, adjust the cumulative number of lanes for each depth by rounding down or up to the nearest multiple of lanes in an aisle.

For example, assume 20 lanes fit along one side of the aisle and the number lanes for the current depth, equal to 8 stacks, is 24 and the the required number of lanes for the next depth, equal to 16, is 14. The number of lanes of the current depth will be rounded down from 24 to 20 and 4 lanes will be carried over to the next depth. 4 lanes of depth 8 hold 32 stacks, this is equivalent to 2 lanes of depth 16. Hence, the required number of lanes of depth 16 is now 16. This number will be rounded up to 20 and -4 lanes carried to the next depth. The result is one side of the aisle has 20 8-deep lanes and the other side has 20 16-deep lanes.

3) Store the products based on their pattern lane depth in the aisles. If no lane of the exact depth is free, pick in an alternating way the next longer or shorter lane to store the product. The actual storage location can be selected from the available empty lanes of the desired length based on handling characteristics such as length of stay of the lane. In other words, lanes that are vacated quicker can be located closer to the Input/Output point.

Observe that the space utilization is only affected by the length of the lane in which the product is stored, not by where this lane is located in the warehouse. Incorporating the material handling costs in the lane selection seems to be a fertile area of further research.

7. CONCLUSIONS

For the single product case, both for the single lane depth and the multiple lane depths cases, efficient algorithms exist to derive the optimal, discrete lane depths. These algorithms are so fast that they can be implemented on microcomputers in real time. For both cases the continuous approximation formulas are sufficiently accurate and even faster to implement.

For most practical situations only a limited number of lane depths are available because of the warehouse layout. The pattern algorithm selects the optimal lane depths from the available lane depths. This algorithm is very efficient and generates solutions which are very close to the unrestricted optimum. This is the only practical algorithm if multiple products have to be stored in the warehouse.

Based on the pattern algorithm and the principle of a perfectly balanced warehouse a set of layout and operating procedures has been derived which will operate the warehouse near maximum space utilization.

The extensive simulation of the proposed policies and the adaptation of the policies to the case where both storage and handling considerations are important are topics for further research.

8. REFERENCES

Aho A. V., Hopcroft J. E. and J. D. Ullman, (1983). Data Structures and Algorithms. Addison-Wesley, Reading, Massachusetts.

Ashayeri J. and Gelders L. F., (1985). Warehouse Design Optimization. European Journal of Operations Research. Vol. 21, pp 285-294.

Berry J. R., (1968). Elements of Warehouse Layout. International Journal of Production Research, Vol. 7, No. 2, pp 105-121.

Goetschalckx M and Ratliff H. D., (1983). Shared Storage Policies. Production and Distribution Research Center Report 83-11, Georgia Institute of Technology, Atlanta, Georgia.

Kind D. A., (1965). Measuring Warehouse Space Utilization. Transportation and Distribution Management, Vol. 7, No. 5, pp 23-33.

Kooy E. D., (1981). Making Better Use of Available Warehouse Space. Industrial Engineering, Vol. 13, No. 10, pp 26-30.

Marsh W. H., (1979). Elements of Block Storage Design. International Journal of Production Research, Vol. 17, No. 4, pp 377-394.

Matson J. O. and White J. A., (1981). Storage System Optimization. Production and Distribution Research Center Report 81-09, Georgia Institute of Technology, Atlanta, Georgia.

Matson J. O. and White J. A., (1984). Modeling Block Stacking Storage Systems. Production and Distribution Research Center Report 84-07, Georgia Institute of Technology, Atlanta, Georgia.

Matson J. O., (1982). The Analysis of Selected Unit Load Storage Systems. Unpublished Doctoral Dissertation, Georgia Institute of Technology, Atlanta, Georgia.

END
12-87
DTIC